

Regenerating Codes for Errors and Erasures in Distributed Storage

K. V. Rashmi, Nihar B. Shah, Kannan Ramchandran, *Fellow, IEEE*, and P. Vijay Kumar, *Fellow, IEEE*

Abstract—Regenerating codes are a class of codes proposed for efficient repair of failed nodes in distributed storage systems. In this paper, we address the fundamental problem of handling errors and erasures which may occur during data reconstruction and node repair in regenerating codes. There are numerous scenarios which motivate this problem such as time-critical data recovery, dynamic load balancing, and security from malicious adversaries. We provide outer bounds, and explicit regenerating codes achieving these bounds for a wide range of system parameters. This also establishes the capacity of these systems for these parameter regimes.

I. INTRODUCTION

Distributed storage systems play a vital role in today's age of big data. For cost considerations, these storage systems often employ commodity hardware, which makes failures a norm rather than an exception. In order to safeguard the precious data against such failures, the data is typically stored in a redundant manner. In this paper, we consider a distributed storage system consisting of n storage nodes in a network, each having a capacity to store α symbols over a finite field \mathbb{F}_q . Data comprising B symbols (the *message*) is to be stored across these n nodes. An end-user (called a *data collector*) must be able to *reconstruct* the entire message by downloading the data stored in *any* k of these n nodes. It follows that such a system can tolerate failure of any $(n - k)$ nodes, and under solely this requirement, can be realised using any $[n, k]$ MDS code.

A second important aspect of a distributed storage system is the handling of node failures. When a storage node fails, it is replaced by a new, empty node. The replacement node is required to obtain the data that was previously stored in the failed node, by downloading data from the remaining nodes in the network. We will term this process as *repair* or *regeneration* of a node. A typical means of accomplishing this is to download the entire message from the network, and extract the desired data from it. However, downloading the entire message, when it eventually stores only a fraction $\frac{1}{k}$ of it, is clearly wasteful of the network resources.

'Regenerating codes' [1] are a class of codes that aim to reduce the amount of download during regeneration, while retaining the storage efficiency of traditional MDS codes. Under the operation of a regenerating code, a replacement node connects to *any* d existing nodes (termed *helper nodes*), and downloads β symbols from each. This setting is illustrated

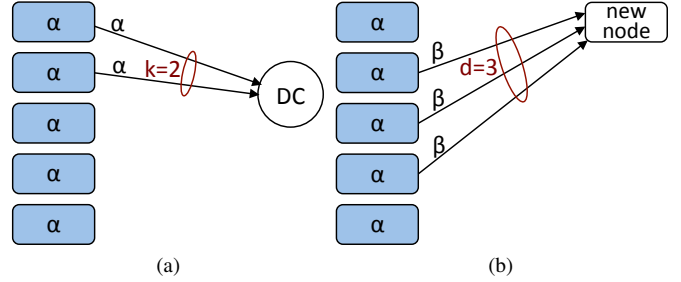


Fig. 1: An example of the system parameters under a regenerating code (in the absence of errors/erasures). The system comprises $n = 5$ storage nodes, data-reconstruction is possible from any $k = 2$ nodes, and node-regeneration from any $d = 3$ nodes.

in Fig. 1. Here, the total amount of data $d\beta$ downloaded for regeneration is much smaller than the total size of the message B . Further, it is shown in [1] that the parameters associated with a regenerating code must necessarily satisfy¹

$$B \leq \sum_{i=0}^{k-1} \min(\alpha, (d-i)\beta) . \quad (1)$$

A regenerating code is said to be *optimal* if it satisfies this bound with equality. Since both storage and bandwidth come at a cost, it is naturally desirable to minimize both α as well as β . However, it can be deduced (see [1]) that achieving equality in (1), for fixed values of B and $[n, k, d]$, leads to a tradeoff between the storage space α and the amount of download for regeneration $d\beta$. The two extreme points in this tradeoff are termed the minimum storage regenerating (MSR) and minimum bandwidth regenerating (MBR) points. These points have been well studied in the literature, and several explicit constructions of codes operating at these points are available [2]–[8]. It has also been shown in [8] that essentially all other points on the tradeoff curve are not achievable.

The amount of data downloaded to regenerate the data stored in a single node is of interest in several other scenarios. For instance, read requests to a temporarily unavailable storage node can be handled by regenerating the data stored in that node: such a scheme is indeed employed in RAID enabled Hadoop Distributed File Systems. The node regeneration property can also be employed to disseminate data in a peer-to-peer content distribution network (see, for example, [9]).

A second example is in handling of data hotspots efficiently, by provisioning new nodes, and treating transfer of the data

K. V. Rashmi, Nihar B. Shah and Kannan Ramchandran are with the Dept. of EECS, University of California, Berkeley, CA 94703, USA. Email: {rashmikv, nihar, kannanr}@eecs.berkeley.edu. P. Vijay Kumar is with the Dept. of ECE, Indian Institute Of Science, Bangalore, India. Email: vijay@ece.iisc.ernet.in. P. Vijay Kumar is also an adjunct faculty member of the Electrical Engineering Systems Department at the University of Southern California, Los Angeles, CA 90089-2565.

¹The authors in [1] consider 'functional' repair where a replacement node only needs to satisfy further reconstructions and repairs. A practical, but more stringent requirement is that of 'exact' repair where it stores data identical to that in the failed node. Throughout this paper, we consider only exact repair.

to the new nodes as node regeneration.

In this paper, we address the problem of handling errors and erasures in distributed storage networks using regenerating codes. In particular, we are interested in codes that can perform data reconstruction and efficient node-regeneration in the presence of errors and erasures at the nodes or in the links. There are numerous practical scenarios which motivate this problem, two of which we discuss below.

Scenario 1 (Seamless reconfiguration): Consider regeneration of a node, accomplished by downloading data from a subset of d other helper nodes. In the midst of this operation, suppose one of these d nodes becomes unavailable (for example, due to a busy server at that node, or a bad communication channel). The replacement node would want to switch to another helper node instead. However, in general, under a (exact) regenerating code, the data passed by a helper node is a function of the identity of the $(d-1)$ other helper nodes (see [10]). This would render the data already downloaded from the other $(d-1)$ nodes useless. Moreover, the reconfiguration would entail additional communication overheads with the other helper nodes informing them of the change. It is thus of interest to construct codes that allow seamless reconfiguration of the helper nodes, allowing replacement of *bad* nodes without additional downloads and overheads.

This property can be modelled as the ability of the regenerating code to handle erasures on the transmission links, as described subsequently in Section II.

Scenario 2 (Security): Consider regeneration of data stored in a node when a subset of the nodes in the system could be compromised to malicious adversaries. In this case, all symbols passed by the compromised nodes contain (adversarial) errors, and need to be corrected for. Moreover, one may also wish to detect errors in order to trace the location of the adversaries.

The aspect of security in distributed storage systems employing regenerating codes is studied in [11] where the authors provide an outer bound for the total amount of data that can be stored securely in the presence of malicious adversaries. They also show that the code constructed in [8] achieves this bound with an appropriate choice of the underlying MDS code, for the case $d = n - 1$ at the MBR point. However, apart from this case, no other constructions of secure regenerating codes are known in the literature.

While we were writing this paper, we came across a contemporaneous work done independently [12] that is related to the present paper, and deals with byzantine fault tolerance using Product-Matrix codes [2]. The authors use a CRC to check the integrity of data during regeneration and reconstruction, and a feedback scheme to iteratively correct them. However, CRC based schemes are not applicable in certain settings such as protection against malicious adversaries, since the CRC can also be corrupted by the adversary.

In the present paper, we adopt a fundamental approach to the handling of errors and erasures in regenerating codes. We consider a system model which considers occurrences of errors and erasures in distributed storage systems. We present outer bounds on the storage and bandwidth requirements under this setting. We also provide explicit code constructions achieving

these bounds for the parameters (i) MSR, all $[n, k, d \geq 2k-2]$ and (ii) MBR, all $[n, k, d]$. This establishes the capacity of such systems for these parameters. Moreover, as a special case, we also establish the capacity of regenerating codes in the presence of malicious adversaries (as considered in [11]) for these parameters, which had remained open. The decoding algorithms have a complexity identical to that of Reed-Solomon codes. The codes presented here are based on a ‘Product-Matrix’ construction introduced in our previous work [2].

The rest of the paper is organized as follows. The system model considered in the paper is described in Section II, and outer bounds for this model are also provided in this section. Explicit constructions of error-resilient regenerating codes are provided in Section III.

II. SYSTEM MODEL

We consider a block-based model where the message is divided into blocks, and there is no coding across the blocks. All operations of encoding, decoding and regeneration are performed independently across the blocks. Thus the regenerating code parameters (for the error-free case) described in Section I can be considered as pertaining to a single block of data. More concretely, we consider a block to consist of B message symbols, and the storage capacity in each of the n nodes to be α symbols per block. One can reconstruct the B -message symbols by downloading the data pertaining to this block from any subset of k nodes, and regenerate the data stored in any node by downloading β symbols each (pertaining to this block) from any d nodes.

We further assume that the granularity of any error or erasure is one block. In other words, we assume that during node-regeneration, all the β symbols passed by a helper node suffer the same fate, i.e., either they are all erased, or all are in error, or all are perfectly received. An identical assumption is made for the α symbols passed by any node during reconstruction, i.e., all the α symbols are assumed to suffer the same fate.

The reason for this assumption is to disallow arbitrarily scattered errors in the model, since codes attempting to guard against such errors would require significantly larger overheads. Moreover, the block-error assumption accurately models several scenarios of interest, when the size of the block is chosen judiciously, as illustrated below.

Consider the scenario of seamless reconfiguration discussed in Section I. The size of a packet during reconstruction and regeneration can be assumed to be a multiple of α and β (since the packet size will usually be much larger than these parameters). Thus when a packet is delayed or dropped all the β (or α) symbols corresponding to a block can be considered as erased.

In the security scenario, to account for compromise of any node or link to a malicious adversary, one needs to protect against corruption of possibly the entire data on that node or link (see [11]: ‘omniscient adversary’). Thus, we can model this scenario under our framework by simply considering the entire data as a single block.

We now define formally, the error/erasure handling capability of a regenerating code.

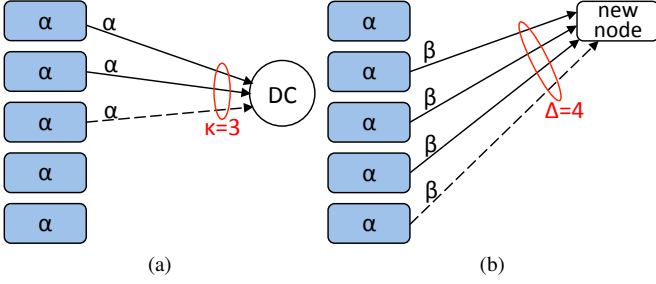


Fig. 2: An example of the system parameters under a resilient regenerating code. The code depicted is $(s = 1, t = 0)$ resilient, i.e., can correct upto of one erasure during reconstruction and node-regeneration. To facilitate this, a connectivity of $\kappa = 3 = k + 1$ is allowed during reconstruction, and of $\Delta = 3 = d + 1$ during regeneration.

Definition 1 ((s, t) -resilient code): A regenerating code is (s, t) -resilient if it can correct upto s erasures and t errors during regeneration as well as reconstruction.

To operate under a error/erasure-prone setting, it is clear that one needs to download additional data. One way to obtain more data during regeneration and reconstruction is to connect to more number of nodes, and we choose this approach. Precisely, we allow an additional connectivity of e , and denote the number of nodes connected to during regeneration as $\Delta = d + e_1$, and during reconstruction as $\kappa = k + e_2$. The parameters e_1 and e_2 depend on the error/erasure correcting capability expected out of the system. Fig. 2 illustrates the setting for a $(1, 0)$ -resilient regenerating code.

We now provide a bound on the capacity of resilient regenerating codes. It is easy to see that the bound provided in [11] for security against ‘omniscient adversaries’ is also applicable to general errors in our block-based setting.² The following theorem extends it to the case of erasures and errors.

Theorem 1: A (s, t) -resilient regenerating code, connecting to Δ and κ nodes for regeneration and reconstruction respectively, must satisfy

$$B \leq \sum_{i=0}^{k-1} \min(\alpha, (d-i)\beta) . \quad (2)$$

where $d = \Delta - s - 2t$ and $k = \kappa - s - 2t$.

Proof (sketch): The bound can be derived either using cut-set arguments in an information flow graph as in [11] or using information theoretic arguments as in [8]. Details are omitted due to lack of space. ■

Remark 1: Observe that this bound is identical to that in (1), for the choice of $e_1 = e_2 = s + 2t$.

We will call an (s, t) -resilient regenerating code as optimal if it meets the bound in Theorem 1. In the next section we present explicit constructions of optimal (s, t) -resilient regenerating codes. These codes meet the bound in Theorem 1, and hence satisfy (2) with equality, and have $\Delta = d + s + 2t$ and $\kappa = k + s + 2t$.

²The notation k , d and b in [11] correspond to κ , Δ and t respectively in this paper.

An appealing feature of these codes is that they are resilient simultaneously for all values of s and t . This feature is particularly useful in the security scenario since it obviates the need of predicting the possible extent of compromise beforehand.

Before moving on to the constructions, we briefly digress to explore some connections with network coding.

Relation to Network Coding: The regenerating codes problem described above, if relaxed to the requirement of regeneration of only the systematic nodes, turns out to be a non-multicast network coding problem. While the occurrence of errors and erasures in multicast network coding are well studied in the literature [13]–[15], the results of the present paper lead to a class of non-multicast networks for which the error/erasure capacity of the network can be achieved by codes that are *linear*, explicit, and deterministic. Moreover, the requirement of (exact) regeneration of the parity nodes, which store functions of the message, presents us with additional constraints as compared to the traditional network coding paradigm. The codes presented in this paper also optimally handle these additional requirements.

III. ERROR-RESILIENT REGENERATING CODES

We provide explicit constructions for error-resilient MSR and MBR codes for

- 1) MSR, all parameters $[n, k, d \geq 2k - 2]$, and
- 2) MBR, all parameters $[n, k, d]$,

which meet the outer bound provided in Theorem 1. Thus, this also establishes the capacity of such a system for these parameter values. These codes are based on Product-Matrix (PM) codes that were introduced in our previous work [2]. Each of the constructions presented are (s, t) -resilient for all values of s and t simultaneously.³

We begin with a description and code constructions for the minimum storage setting, and subsequently present the minimum bandwidth case. In both cases, we first briefly describe the Product-Matrix code construction for the error/erasure-free case [2], and then prove its error and erasure correcting capability.

A. (s, t) -resilient MSR Codes

MSR codes use the minimum possible storage at each node. Since a data-collector connecting to any k nodes should be able to reconstruct all the B message symbols, each node must necessarily store at-least a fraction $\frac{1}{k}$ of the entire data. Hence for an MSR code we have $\alpha = \frac{B}{k}$. To meet the bound in (1) with equality (in absence of errors/erasures), an MSR code must satisfy

$$B = k\alpha, \quad d\beta = \alpha + (k-1)\beta . \quad (3)$$

In this section we present explicit constructions of MSR codes for all parameter values $[n, k, d \geq 2k - 2]$ which are (s, t) -resilient, for all s and t simultaneously. An example of a $(1, 0)$ -resilient MSR code is provided in Fig. 3.

³Provided, of course, that desired connectivity is available, i.e., $\Delta \leq n - 1$, $\kappa \leq n$

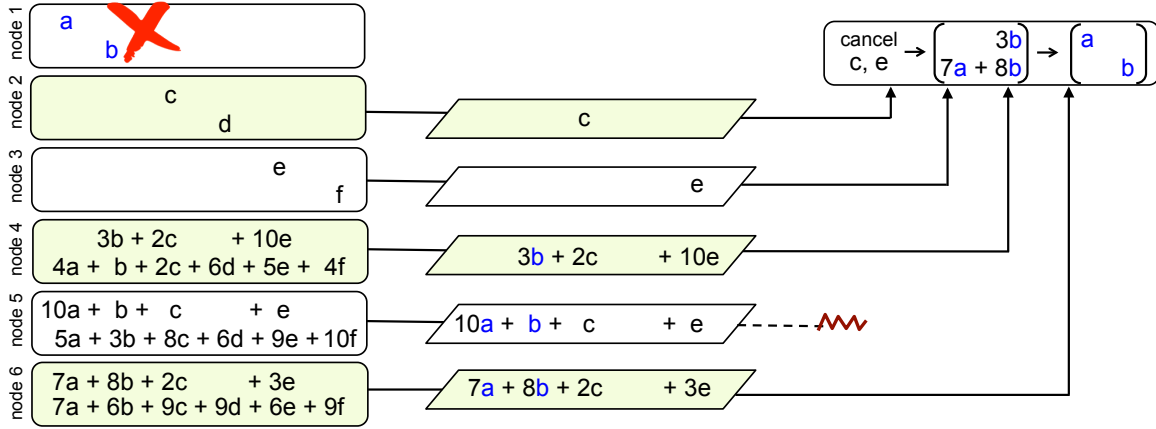


Fig. 3: An example of a $(1, 0)$ -resilient MSR code with parameters $n = 6$, $B' = 6$, $\alpha' = 2$, $\beta' = 1$. Also depicted is an instance of one erasure correction by connecting to $\Delta = d + 1$ nodes during regeneration of the data stored in node 1.

The code is designed for the case $d = 2k - 2$, and this can be extended to any $d > 2k - 2$ via shortening technique for MSR codes provided in [2], [4]. For the case of $d = 2k - 2$, from (3) we get

$$\alpha = (k - 1)\beta, \quad B = k(k - 1)\beta. \quad (4)$$

Since α and B both are multiples of β , we obtain the optimal code for the desired parameters (B, α, β) by first constructing an optimal code for the case

$$\alpha' = (k - 1), \quad B' = k(k - 1) = \alpha'(\alpha' + 1) \quad \beta' = 1, \quad (5)$$

and then concatenating this code β times in parallel.

The PM-MSR code in [2] can be described in terms of an $(n \times \alpha')$ code matrix $C = \Psi M$, with the i^{th} row of C containing the α' symbols stored in node i . The $(n \times d)$ encoding matrix Ψ is of the form $\Psi = [\Phi \quad \Lambda\Phi]$, where Φ is an $(n \times \alpha')$ matrix and Λ is an $(n \times n)$ diagonal matrix satisfying: (a) any α' rows of Φ are linearly independent, (b) any d rows of Ψ are linearly independent, and (c) the diagonal elements of Λ are all distinct. The choice of the matrix Ψ governs the choice of the finite field \mathbb{F}_q , e.g., choosing Ψ as Vandermonde (carefully chosen to satisfy the condition on Λ) permits any $q \geq 4n$. The $((d = 2\alpha') \times \alpha')$ message matrix M is of the form $M = [S_1 \quad S_2]^t$, where S_1 and S_2 are $(\alpha' \times \alpha')$ symmetric matrices. The superscript t is used to denote the transpose of a vector or matrix. The two matrices S_1 and S_2 together contain $\alpha'(\alpha' + 1)$ distinct symbols, and these positions are populated by the $B = \alpha'(\alpha' + 1)$ message symbols.

The following theorems show that this code is (s, t) -resilient during regeneration as well as reconstruction, for all values of s and t .

Theorem 1 (Node-Regeneration): In the MSR code presented, the α symbols stored in any node can be regenerated by downloading β symbols each from any other $\Delta = d + s + 2t$ nodes, in the presence of upto s (block) erasures and t (block) errors.

Proof: Since we consider only block errors and erasures, it is sufficient to describe the regeneration algorithm for the code with $\beta' = 1$, and the same algorithm is applied in parallel to obtain the regeneration algorithm for the desired code.

Consider failure of node f in the system, and let $[\phi_f^t \quad \lambda_f \phi_f^t]$ be the row of Ψ corresponding to the failed node. Thus the α' symbols stored in node f are

$$[\phi_f^t \quad \lambda_f \phi_f^t] M = \phi_f^t S_1 + \lambda_f \phi_f^t S_2. \quad (6)$$

The replacement for the failed node f connects to an arbitrary set $\{h_j \mid j = 1, \dots, \Delta\}$ of Δ nodes. To facilitate regeneration of the data of node f , node h_j computes the inner product $\psi_{h_j}^t M \phi_f^t$ and passes on this value to the replacement node. Letting $\underline{m}_f = M \phi_f^t$, we can write the symbol passed by node h_j as $\psi_{h_j}^t \underline{m}_f$. Thus the Δ symbols obtained at the destination are $\Psi_{\text{reg}} \underline{m}_f$, where

$$\Psi_{\text{reg}} = [\psi_{h_1}^t \quad \psi_{h_2}^t \quad \dots \quad \psi_{h_\Delta}^t]^t.$$

Since any d rows of Ψ are linearly independent by construction, and since Ψ_{reg} comprises a subset of the rows of Ψ , $\Psi_{\text{reg}} \underline{m}_f$ is simply an MDS encoding of the d symbols in the vector \underline{m}_f . It follows that this code has a minimum distance of $\Delta - d + 1 = s + 2t + 1$ which allows us to recover \underline{m}_f using standard decoding algorithms [16] in the presence of upto s erasures and t errors. Thus the replacement node now has access to

$$\underline{m}_f = M \phi_f^t = \begin{bmatrix} S_1 \phi_f^t \\ S_2 \phi_f^t \end{bmatrix}.$$

Since S_1 and S_2 are symmetric matrices, the replacement node has equivalently acquired $\phi_f^t S_1$ and $\phi_f^t S_2$. Using this it can obtain

$$\phi_f^t S_1 + \lambda_f \phi_f^t S_2, \quad (7)$$

which is precisely the data previously stored in node f . ■

Theorem 2 (Data-Reconstruction): In the MSR code presented, a data-collector can reconstruct all the B message symbols by downloading data stored in any $\kappa = k + s + 2t$ nodes in the presence of upto s (block) erasures and t (block) errors.

Proof (sketch): The data reconstruction property of the code in the error-free case, as shown in [2], implies that the data passed by the κ nodes are MDS over the finite field

\mathbb{F}_q^α . Over this finite field, the message is of size k , and the minimum distance of this MDS code is $\kappa - k + 1 = s + 2t + 1$. This guarantees reconstruction of the k source symbols over \mathbb{F}_q^α , and equivalently the $k\alpha = B$ source symbols over \mathbb{F} , in the presence of upto s erasures and t errors.

We omit the details on explicit decoding algorithms due to space constraints. ■

B. (s, t) -resilient MBR Codes

MBR codes achieve the minimum possible download during regeneration: a replacement node downloads only what it stores, resulting in $d\beta = \alpha$. To meet the bound in (1) with equality (in absence of errors/erasures) an MBR code must satisfy

$$B = \left(kd - \binom{k}{2} \right) \beta, \quad \alpha = d\beta. \quad (8)$$

In this section we present explicit constructions of MBR codes for all parameter values $[n, k, d]$ which are (s, t) -resilient, for all s and t simultaneously. As in the case of MSR codes, B and α are multiples of β , and we first construct codes for

$$B' = \left(kd - \binom{k}{2} \right), \quad \alpha' = d, \quad \beta' = 1. \quad (9)$$

The desired code can be obtained by concatenating β copies of this code.

The PM-MBR code in [2] has an identical form $C = \Psi M$ as PM-MSR code. The MBR code has the $(n \times d)$ encoding matrix Ψ of the form $\Psi = [\Phi \quad \Sigma]$, where Φ is an $(n \times k)$ matrix satisfying: (a) any k rows of Φ are linearly independent, (b) any d rows of Ψ are linearly independent. For instance, one can choose Ψ to be a Vandermonde matrix. The $(d \times d)$ message matrix M is symmetric and consists of the B' message symbols arranged in the following manner:

$$M = \begin{bmatrix} S & T \\ T^t & 0 \end{bmatrix}.$$

Here, the $((d - k) \times k)$ matrix T and the $(k \times k)$ symmetric matrix S contain the $B' = kd - \binom{k}{2} = k(d - k) + \frac{k(k+1)}{2}$ message symbols as their elements.

The following theorems show that this code is (s, t) -resilient during regeneration as well as reconstruction, for all values of s and t .

Theorem 3 (Node-Regeneration): In the MBR code presented, the α symbols stored in any node can be regenerated by downloading β symbols each from any $\Delta = d + s + 2t$ nodes, in the presence of upto s (block) erasures and t (block) errors.

Proof: As in the case of MSR, it is sufficient to describe the regeneration algorithm for the code with $\beta' = 1$. Consider failure of node f in the system, and let ψ_f^t be the row of Ψ corresponding to the failed node. Thus the α' symbols stored in node f are $\psi_f^t M$. We will follow the notation as in Theorem 1. The helper node h_j passes the symbol $\psi_{h_j}^t M \psi_f$. Denoting $\underline{m}_f = M \psi_f$, the Δ symbols obtained at the destination can be written as $\Psi_{\text{reg}} \underline{m}_f$ where

$$\Psi_{\text{reg}} = \begin{bmatrix} \psi_{h_1} & \psi_{h_2} & \dots & \psi_{h_\Delta} \end{bmatrix}^t.$$

By construction, $\Psi_{\text{reg}} \underline{m}_f$ corresponds to MDS encoding of the vector \underline{m}_f . As in the case of MSR, this code has minimum distance of $\Delta - d + 1 = s + 2t + 1$ which allows us to recover \underline{m}_f in the presence of upto s erasures and t errors. Since the message matrix M is symmetric, \underline{m}_f is precisely the α' symbols required. ■

Theorem 4 (Data-Reconstruction): In the MBR code presented, a data-collector can reconstruct all the B message symbols by downloading data stored in any $\kappa = k + s + 2t$ nodes in the presence of upto s (block) erasures and t (block) errors.

Proof (sketch): As in the MSR case, the proof exploits the data-reconstruction property of PM-MBR codes in the error-free case [2]. The reconstruction property implies that, over \mathbb{F}_q^α , the minimum distance of the code is $\kappa - k + 1 = s + 2t + 1$. This guarantees reconstruction of the k symbols over \mathbb{F}_q^α , and equivalently the B source symbols over \mathbb{F}_q , in the presence of upto s erasures and t errors. Again, we omit details on explicit decoding algorithms due to space constraints. ■

ACKNOWLEDGEMENT

The authors would like to thank Salim El Rouayheb and Sameer Pawar of UC Berkeley for fruitful discussions.

REFERENCES

- [1] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," *IEEE Trans. Inf. Theory*, vol. 56, no. 9, pp. 4539–4551, 2010.
- [2] K. V. Rashmi, N. B. Shah, and P. V. Kumar, "Optimal exact-regenerating codes for the MSR and MBR points via a product-matrix construction," *IEEE Trans. Inf. Theory*, vol. 57, no. 8, pp. 5227–5239, Aug. 2011.
- [3] K. V. Rashmi, N. B. Shah, P. V. Kumar, and K. Ramchandran, "Explicit construction of optimal exact regenerating codes for distributed storage," in *Proc. Allerton Conf.*, Urbana-Champaign, Sep. 2009.
- [4] N. B. Shah, K. V. Rashmi, P. V. Kumar, and K. Ramchandran, "Explicit codes minimizing repair bandwidth for distributed storage," in *Proc. IEEE ITW*, Cairo, Jan. 2010.
- [5] C. Suh and K. Ramchandran, "Exact regeneration codes for distributed storage repair using interference alignment," in *Proc. ISIT*, Austin, Jun. 2010.
- [6] I. Tamo, Z. Wang, and J. Bruck, "Mds array codes with optimal rebuilding," in *Proc. IEEE ISIT*, St. Petersburg, Jul. 2011.
- [7] D. Papailiopoulos, A. Dimakis, and V. Cadambe, "Repair optimal erasure codes through hadamard designs," in *Proc. Allerton Conf.*, 2011.
- [8] N. B. Shah, K. V. Rashmi, P. V. Kumar, and K. Ramchandran, "Distributed storage codes with repair-by-transfer and non-achievability of interior points on the storage-bandwidth tradeoff," *IEEE Trans. Inf. Theory*, vol. 58, no. 3, 2012.
- [9] S. Pawar, S. Rouayheb, H. Zhang, K. Lee, and K. Ramchandran, "Codes for a Distributed Caching based Video-On-Demand System," in *Asilomar Conference on Signals, Systems, and Computers*, 2011.
- [10] Y. Wu, A. G. Dimakis, and K. Ramchandran, "Deterministic regenerating codes for distributed storage," in *Proc. Allerton Conf.*, Sep. 2007.
- [11] S. Pawar, S. El Rouayheb, and K. Ramchandran, "Securing dynamic distributed storage systems against eavesdropping and adversarial attacks," *IEEE Trans. Inf. Theory*, vol. 57, no. 10, pp. 6734–6753, 2011.
- [12] Y. Han, R. Zheng, and W. Mow, "Exact regenerating codes for byzantine fault tolerance in distributed storage," in *Proc. INFOCOM*, 2012.
- [13] N. Cai and R. Yeung, "Network error correction, ii: Lower bounds," *Comm. in Information & Systems*, vol. 6, no. 1, pp. 37–54, 2006.
- [14] S. Jaggi, M. Langberg, S. Katti, T. Ho, D. Katabi, and M. Médard, "Resilient network coding in the presence of byzantine adversaries," in *INFOCOM*, 2007.
- [15] R. Koetter and F. Kschischang, "Coding for errors and erasures in random network coding," *IEEE Trans. Inf. Theory*, vol. 54, no. 8, pp. 3579–3591, 2008.
- [16] F. MacWilliams and N. Sloane, *The Theory of Error-Correcting Codes, Part I*. North-Holland Publishing Company, 1977.